Downloadable TCP의 설계 및 구현

황재현° 최진희 김세원 유혁
고려대학교 컴퓨터학과
{jhhwang°, jhchoi, swkim, hxy}@os.korea.ac.kr

Design and Implementation of Downloadable TCP

Jae-Hyun Hwang° Jin-Hee Choi Se-Won Kim Hyuck Yoo
Dept. of Computer Science, Korea University

요 약

TCP는 유선 네트워크 환경을 대상으로 설계된 신뢰성 있는 전송 계층 프로토콜이다. 그러나 무선 네트워크와 같이 기존의 설계 가정에서 벗어난 새로운 환경들은 TCP로 하여금 성능의 저하 문제를 야기하였으며, 이를 극복하기 위한 여러 가지 변종들이 제시되어 왔다. 그러나 하나의 단말 내에서 모든 상황에 적합한 TCP 변종을 모두 유지하는 것은 어려운 일이며, 이를 해결하기 위한 방법으로 제시된 것이 바로 모바일 코드를 이용한 프로토콜의 다운로드이다. 이와 관련된 기존의 연구들은 전체 소스 코드를 다운받아 컴파일한 뒤 갱신하는 방법을 취하고 있는데 이는 TCP의 부분 갱신을 위한 효율적인 프레임워크가 없기 때문이다. 이에 본 논문에서는 전체 프로토콜이 아닌 필요한 부분만을 쉽게 갱신 가능한 TCP 프레임워크를 제안하며, 그 설계와 구현에 대해 다루고자 한다.

1. 서 론

오늘날 통신 기술의 발전은 다양한 형태의 네트워크가 등장하게 하였고, 이는 사용자로 하여금 서비스 선택의 폭을 넓히게 하였다. 이와 더불어 단말의 이동성은 각 환경에 적합한 프로토콜의 동적인 구성을 요구하고 있다. 이러한 서비스 패러다임의 변화는 프로토콜의 적응성과 확장성에 대한 연구를 지속적으로 이끌어왔다. 대표적으로 TCP(Transmission Control Protocol)를 들 수 있으며, 최근까지도 다양한 상황에 적응성 있는 TCP 변종들이 연구되고 있다.

그러나 하나의 단말이 모든 상황에 적합한 프로토콜을 모두 유지하는 것은 어려운 일이다. 이 문제를 해결하기 위해 기존의 연구에서 제시되었던 방법은 필요한 프로토콜을 모바일 코드 형태로 다운로드 받는 것이었다. 즉, 전체 프로토콜을 소스 코드 수준으로 다운로드 받아 이를 컴파일하고 갱신하는 과정을 취하고 있다[1][2]. 이 방법은 구현이 쉽다는 장점을 가지나, 다음과 같은 몇 가지 문제점을 포함한다. 먼저 전체 프로토콜을 다운로드 받기 때문에 모바일 코드의 크기가 커지며, 이는 다운로드 시간을 증가시킨다. 또한 소스 수준의 코드이기 때문에 컴파일 시간이 추가되며 전체적으로 갱신 시간이 너무 길어진다는 단점을 갖는다. 다운로드 시간을 단축시키기 위해 소스 코드를 압축하는 방법이 제안되었지만 압축을 풀기 위한 시간이 추가되므로 여전히 갱신 시간은 줄어들지 않는다. 특히, 대부분의 경우 프로토콜의 전체가 아닌 일부분의 갱신만으로 충분하기 때문에 비효율적이라 할 수 있다.

이와 같은 단점에도 불구하고 기존의 연구들이 소스 수준의 갱신을 가정해온 이유는 바이너리 수준의 부분적인 갱신을 지원하는 효율적인 프레임워크가 없었기 때문이다. 이에 본 논문에서는 부분적인 바이너리 갱신이 가능한 구조를 갖는 Downloadable TCP(D-TCP)를 제안한다. 제안된 방식은 TCP의 상태 기계(State Machine) 구조를 따르며, 각각의 상태들을 단위로 TCP의 기능을 구현하고 갱신하도록 하고 있다. 본 논문의 초점은 먼저 이와 같은 프레임워크를 바탕으로 TCP를 직접 구현하고 제대로 동작하는지를 검증하는데 있으며, 이를 바탕으로 차후 프로토콜의 부분적인 갱신에 대한 연구 진행 방향에 대해 언급하기로 한다.

2. D-TCP 프레임워크와 컴포넌트의 구조

TCP는 기본적으로 상태 기계(State Machine) 다이어그램에 의해 그 작동 메커니즘이 설명된다. 그러나 대부분의 TCP 구현들은 성능 상의 이유로 상태 기계로써 구현되기보다 연속적인 함수 호출의 구조를 가지고 있다. 이와 같은 구조는 기능 간의 구분이 명확하지 않기 때문에 부분 갱신이라는 목표에 적합하지 않으며, 프로토콜의 확장성을 떨어뜨린다. 이에 D-TCP는 성능에 앞서 부분 갱신에 초점을 맞추었으며, 이를 위해 기존의 구현 방

본 연구는 한국과학재단 특정기초연구(R01-2004-000-10588-0) 지원으로 수행되었음.

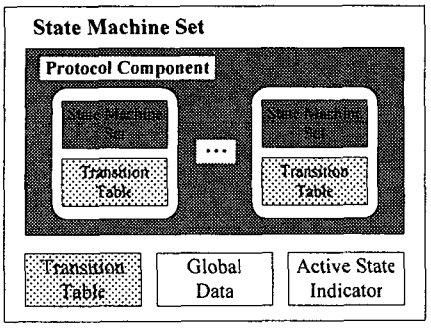

그림 1. Downloadable TCP 프레임워크와 컴포넌트의 전체 구조

식이 아닌 상태 기계의 구조로 설계하였다.

D-TCP의 구현 모델은 크게 프로토콜 프레임워크와 프로토콜 컴포넌트로 나누어진다.(그림 1) 프로토콜 프레임워크는 전체 상태 기계를 포함하며, 상태 간의 전이 테이블(Transition Table), TCP의 TCB (Transmission Control Block)와 같은 전역 데이터(Global Data), 그리고 현재 활성화 중인 상태를 가리키는 지시자(Active State Indicator) 등으로 구성된다. 프레임워크는 여러 개의 독립적인 컴포넌트를 포함하고 있는데, 각 컴포넌트는 Closed, Listen, Established 등과 같은 TCP의 상태들을 의미한다. 각 상태들은 또 하나의 상태 기계가 될 수 있는데, Established 상태가 Transmit, Retransmit, Persist 등의 하부 상태 기계로 확장되는 경우가 이에 해당한다. 따라서 프레임워크는 전체 상태 기계들의 최상위 추상 단계라 볼 수 있다.

전이 테이블은 이벤트의 발생에 따른 상태 이동 정보를 유지한다. 제안된 모델에서 이벤트는 전역 이벤트와 지역 이벤트로 나뉘는데, 전역 이벤트는 *PKTSENT, PKTRCVD, TIMEOUT* 등으로 프레임워크에 전달되는 것이며, 지역 이벤트는 해당 상태에서 발생되는 것이다. 이러한 이벤트를 통해 각 상태들은 [현재 상태, 이벤트]를 쌍으로 하는 키 값을 생성하고 해당 이벤트에 대해 전이해야 할 상태 기계들을 해시 테이블로 유지한다.

3. D-TCP의 구현

D-TCP는 현재 테스트 및 디버깅의 편의를 위해 유저 레벨의 응용 프로그램 형태로 구현되었으며, 하위 IP 계층은 RAW 소켓 인터페이스를 사용하고 있다.

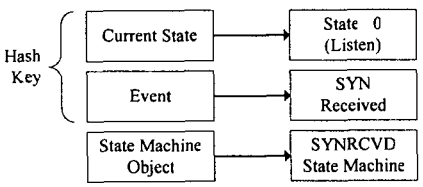

그림 2. 전이 테이블의 예

start() interface

```
//Global Event Handling
switch(G_Event) {
     case PKTSENT: goto PKTOUT;
     case PKTRCVD: goto PKTIN;
                ...
}

PKTOUT:
     /* Outgoing packet processing */
PKTIN:
     /* Incoming packet processing */
EXEC:
     /* Local event handling */
```

D-TCP의 시작은 프레임워크의 활성화 상태가 가리키는 컴포넌트(하나의 상태 기계)부터 이며, 각 컴포넌트들은 프레임워크가 유지하는 전역 데이터에 접근하여 해당 코드를 수행한다. 또한, 부분 갱신과 코드의 배포를 고려하여 인터페이스는 *start(), buildTTable()* 로 통일시켰으며, 각각 해당 상태의 코드 실행, 초기 전이 테이블 구성 시에 사용된다. *start()* 함수는 크게 3부분으로 구성된다. 먼저 함수로 넘어온 전역 이벤트를 확인하여 PKTOUT, PKTIN, TIMEOUT 등의 코드 수행부분으로 점프한다. 해당 상태에서의 패킷 처리가 끝나면 지역 이벤트가 발생하고 EXEC 영역으로 이동하여 각 이벤트에 따라 다른 상태로 전이하거나 이벤트 처리를 수행한다.

그림 3은 3-Way Handshaking이 끝나고 Established 상태가 활성화 중일 때 데이터를 주고 받는 과정을 나타

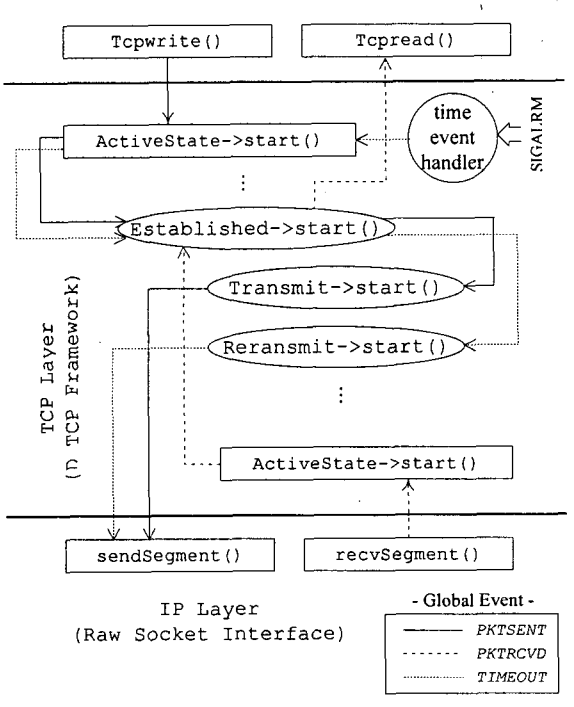

그림 3. Established 상태에서의 D-TCP 실행 과정

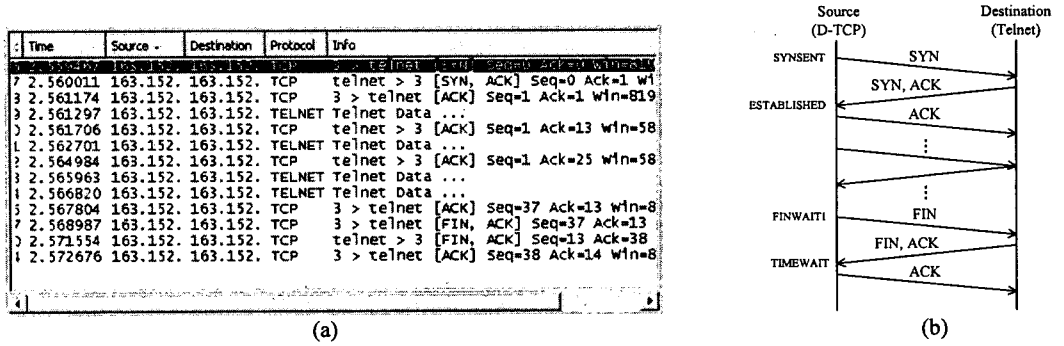

그림 4. (a) Ethereal - 패킷 캡쳐 화면, (b) 패킷 교환에 따른 D-TCP 의 활성화 상태(Active State)

낸 것이다. 언급한 바와 같이, 모든 경우에 있어 제일 먼저 프레임워크의 지시자가 가리키는 활성화 상태의 start() 함수가 호출되며, 전달된 이벤트에 따라 해당 코드가 실행된다. Established 상태는 전송(output)과 관련된 또 하나의 하위 레벨 상태 기계를 포함하고 있는데 이를 위해 별도의 전송상태 지시자를 유지한다. 따라서 데이터를 보낼 때에는 내부적으로 다시 전송상태 지시자가 가리키는 컴포넌트의 *start()* 함수를 호출하게 되며, 일반적인 패킷 전송 시에는 Transmit 상태로, TIMEOUT 이벤트 발생(즉, 재전송) 시에는 Retransmit 상태로 각각 전이한다. 패킷을 받는 경우는 별도의 전이 없이 그대로 Established 상태의 PKTIN 부분이 실행된다.

4. 실험 내용

D-TCP의 본 목적은 프로토콜의 부분적인 기능 갱신에 있지만 그에 앞서 제안한 구현 모델이 제대로 동작하는지를 검증하는 단계가 필요하다. 이를 위해 D-TCP와 다른 호스트에서 서비스하는 텔넷 프로그램과의 통신을 시도하였다. 먼저 D-TCP를 송신 쪽으로 하여 세션을 연결한 후 몇 번의 짧은 데이터를 전송하였으며, 곧바로 세션을 닫는 간단한 테스트 과정을 거쳤다. 여기서 한가지 발생되는 문제는 RAW 패킷의 IP 헤더 내 프로토콜 번호가 255라는 점이다. D-TCP는 현재 통신을 위해 RAW 소켓 인터페이스를 사용하고 있기 때문에 수신 패킷의 프로토콜 번호가 255로 세팅 되어야 응용 프로그램이 받을 수 있게 된다. 반면, 텔넷은 IP 헤더의 프로토콜 번호를 6(TCP)으로 설정하기 때문에 이러한 상황에서는 통신이 이루지지 않는다. 이 문제를 해결하기 위해 D-TCP가 실행 중인 호스트의 리눅스 커널을 일부 수정하였으며, 수신 패킷의 해당 응용 프로그램이 없으면 RAW 패킷으로 간주하여 한 번 더 찾도록 하는 방법을 사용하였다.

그림 4는 Ethereal이란 패킷 캡쳐 프로그램을 사용하여 송수신자 사이의 패킷 교환 과정을 나타낸 것이다. 화면의 내용과 같이 3-Way Handshaking과 데이터의 전송, ACK의 수신, 세션의 종료 과정이 정확하게 수행되었음을 확인할 수 있었으며, 각 상태로의 전이 역시 설계 방향대로 작동했음을 알 수 있다.

5. 결론 및 향후 연구 방향

본 논문에서는 모바일 코드를 통한 프로토콜의 부분적인 갱신을 지원하는 Downloadable TCP를 제안하고 설계, 구현하였다. 기존의 연구들이 소스 수준의 전체 프로토콜을 다운로드 받는데 비해, 제안된 방식은 일부 바이너리 컴포넌트를 다운로드 받아 동적으로 부분 갱신할 수 있는 구조를 갖도록 설계하였으며 전체적으로 프로토콜 갱신 시간을 단축하고자 하였다.

지금까지 진행된 상황은 본 논문에서 제안된 구조가 다른 TCP 구현들과 통신이 가능하며, 정확히 동작한다는 것을 확인한 단계이다. 현재 본 연구는 동적인 부분 갱신에 초점을 맞추어 진행되고 있으며, 핵심 메커니즘은 바로 전이 테이블에 있다. 즉, 프레임워크가 초기에 구성한 전이 테이블 정보를 특정 위치에 독립적으로 등록시킴으로써 테이블 조작만으로 갱신이 가능하게 하는 것이다. 이 후, 다운로드 된 모바일 코드는 제일 먼저 현재의 전이 테이블 정보를 읽어온 후, 교체할 위치를 추적하여 전이할 상태를 자기 자신으로 바꾸게 된다. 이러한 바이너리 코드의 부분 갱신 메커니즘은 앞으로 TCP의 배포 문제에도 크게 기여할 것으로 예상한다.

참고 문헌

[1] P. Patel, D. Wetherall, J. Lepreau, and A. Whitaker, " TCP Meets Mobile Code" , HotOS IX. 2003.

[2] P. Patel, A. Whitaker, D. Wetherall, J. Lepreau, and T. Stack, " Upgrading Transport Protocols using Untrusted Mobile Code" , In Proc. of the 19th ACM Symposium on Operating Systems Principles, 2003.

[3] Douglas E. Comer, David L. Stevens, " Internetworking with TCP/IP volume 2" , Prentice Hall International Editions. 1999.

[4] K. Wehrle, F. Pahlke, H. Ritter, D. Muller, M. Bechler, " The Linux Networking Architecture" , Prentice Hall. 2005.